Towards Full Integration Of XML And Advanced Database Concepts

David Olsen,(E-mail: dolsen@b202.usu.edu), Utah State University
Vance Cooney, (E-mail:vcooney@ewu.edu),Eastern Washington University
Bryan Marshall, (E-mail: bryan.marshall@usu.edu), Utah State University
Richard Swart, (E-mail: richard.swart@business.usu.edu), Utah State University

ABSTRACT

Most advanced database systems courses focus on core aspects of relational design, data modeling, transaction processing, and distributed database issues. Given the ever increasing importance of web enabled databases generally but particularly the influence of XML (eXtensible Markup Language), an alternative approach would be to teach the traditional core principles while integrating an XML module into the course. The focus of this paper is to elaborate on how such an integration would be accomplished in an advanced database course.

INTRODUCTION

Advanced database courses traditionally cover the theory and concepts behind the design and implementation of relational databases. In recent years there has been mounting pressure to weave into this discussion content on integration with the Web as much e-commerce is facilitated by web-enabled databases. The purpose of this paper is to show how we enriched an advanced database class by adding an XML (eXtensible Markup Language, the emerging standard technology for database / web integration) module that extended a typical a real world database example with an integrated XML component. To do this we developed a database using Microsoft's SQL Server 2000, referenced several online tutorials, and created assignments to "pull" data from this database using the XML technology [1].

The remarkable history of the Internet and WWW, well-documented elsewhere, is marked in the mid 1990s by the increasing realization that for the web to facilitate e-commerce in a meaningful way, that production databases and web servers would have to be integrated so that ordering et. al. would be from real time inventories. With the explosion in web sites and web services designed to support this functionality, the limitations in HTML, the web's first language, became obvious [2]. Specifically, many businesses needed to pass data from dissimilar information systems (typically relational databases) via the web and HTML based web pages were not well suited to this requirement. The reason is that such transfers are greatly facilitated by a medium that describes the structure of data, a requirement that vanilla HTML cannot meet. This is where XML comes in.

Prior to the development of XML there were efforts to adapt the precursor of HTML, SGML (Standard Generalized Markup Language) to accommodate the requirements of web – enabled database applications. SGML was a descendant of IBM's Generalized Markup Language (GML), originally developed in the 1960s to enable the sharing of machine-readable documents in large projects in government. It had also been used extensively in the printing and publishing industries, but its complexity had prevented its widespread application for small-scale general-purpose use [cite = http://en.wikipedia.org/wiki/SGML]. So, when efforts to adapt SGML itself proved fruitless, Jon Bosak, Tim Bray, C. M. Sperberg-McQueen, and Jean Paoli of Microsoft, designed a simplified version of XML based on SGML that has since evolved into the standard XML that we have today [3].

So, XML today is not merely an extension of HTML, it is a meta-language that can be used to define a language particular to a business domain and allow the exchange of data using this defined language. For example, instead of just, say, the <H1></H1> tag pair that traditional HTML would offer that only has to do with how the text within tags is displayed, an XML tag has semantic value. An XML tag example might be <Whsle_Unit_Cost></Whsle_Unit_Cost> and such a tag would define the meaning of the data as well as the display

format; i.e., data between the pairs would mean the wholesale unit cost of something, say a prescription drug, and an organization receiving an XML file so designed could easily import the data within into their database systems. In short, XML allows a user to separate the presentation of data from its storage and management. This allows users a vast array of opportunities for using XML to exchange data with trading partners. This phenomenon alone justifies inclusion of XML technology in an advanced database class in the author's opinion.

Since its inception, XML has been adopted by many developers as a way to describe data sets and their contents and to define how the data should be output or displayed on a web page (or, significantly, a cell phone, PDA or any of a number of other human readable or machine readable devices). Before XML, a client application accessing a database needed to translate the result set returned to a format that could be understood and displayed by a web application. XML removed the need for client side processing (given a XML compliant client) as the data and its formatting were defined in the XML markup.

The importance of XML is supported by the fact that Microsoft SQL Server 2000 supports XML, and the result sets can be returned directly in XML format, or data can be retrieved from an XML document as if it were a SQL Server table. A list of XML terms and definitions provided to students in the described database module are included in Appendix A. These terms are a good reference for anyone wishing to begin the study of XML.

BRIEF LITERATURE REVIEW

After Codd's seminal 1970 paper on relational theory, hundreds of books and articles on relational databases have been published [5]. Knowledge of relational database design and normalization are staples of advanced database courses, and are key components of the IS2002 model curriculum and guidelines for undergraduate degree programs in information systems curriculum [6]. Likewise, many computer science departments rely strongly on the Computing Curriculum 2001 [7]. We believe that while model curricula such as IS2002 are extremely valuable guidelines, they are by their nature conservative, and likely to be slow in responding to emerging technologies such as XML. However, we believe that XML is going to be integral to future web enabled database architectures and so believe that universities need to instruct students in XML. As a first pass at such development we have developed an XML Module that can be integrated into an advanced database management course. This will help students prepare to use modern development tools using XML such as Microsoft's .NET framework, and Sun Microsystems's J2EE platform. The need to teach XML in database courses is further demonstrated by the inclusion of XML column types, XML views on relational data, relational views on XML data, XML Schema, and XML based query languages [8] in Microsoft SQL Server 2000 and Oracle 10g. What follows is a description of the teaching module we developed for advanced undergraduates and graduate students in an advanced database management course.

MOAB MEDICAL CLINIC – THE BACKGROUND CONTEXT FOR THE MODULE

One of the challenges of database instruction is creating interesting cases relevant to students. Many of Utah State University students are avid outdoor enthusiasts and our campus in located next to the entrance of a national forest with extensive biking trails. The biking trails around Moab are internationally famous, and the authors believe that students will easily identify with the nature of mountain biking and understand the common injuries from this sport. Many students have visited a clinic or emergency room for an injury at some point in their life, so a clinical model was chosen to illustrate principals of database design and to facilitate the student's comprehension of the queries that are assigned. Their familiarity with the material should assist them to interpret the correctness of the output of their queries and to formulate appropriate queries.

The Moab Medical Clinic (MMC) is located just outside of Moab, Utah near many of the famous slick rock mountain biking trails and off-road vehicle trails. It was primarily designed to handle emergency cases that result from accidents involving mountain bikes and off road vehicles. It was located near many popular sites so that emergency response times could be minimized. Common cases at the clinic include fractured bones, concussions, abrasions, and puncture wounds from these outdoor accidents. Most of the patients entering the clinic will have common injuries and report pain.

The case approaches the design of the clinic's functions using a simplified model of care. Students are not expected to be familiar with medical clinic operations, and this teaching case builds upon common experiences in receiving medical care to determine the entities and relationships in the database.

ENTITY RELATIONSHIP MODELING

Students are given a textual case based on the MMC. The students are to read the case and "ferret out" relevant database information. Students then determine the relevant entities and relationships based on information provided in the case. After the entities and relationships are determined, students create the database while enforcing the key constraints and business rules. A copy of the student assignment is included in Appendix A , the ER Diagram is in Figure 1 and the SQL Relationship Diagram is found in Figure 2.

A patient presents herself and may be admitted to the MMC. In the process of being admitted, her address, date of birth and other identifying information are recorded along with the patient's self-reported symptoms. MMC staff record the patient's injuries and classify them according to a list of standard injuries. If a medication is prescribed, the name, time and dosage amount are recorded. MMC staff then record the patient's response to the medication. Since patients may frequent the clinic, each admission is separately recorded, though all patient and injury information can be collected and aggregated. Billing information and data on payments are collected for each billing cycle.

LOGICAL RELATIONAL MODELING – SPECIFYING TABLES AND FIELDS

The patients table contains tuples for patients. Biographical and contact information about the patient are collected. Each patient receives a unique patient identification number (PatientID) which serves as the primary key for the table.

The admittance table records the facilities tracking of patients that are admitted to the clinic, as opposed to referred to a primary care provider or sent on to a regional medical center. As patients are admitted, an admittance identifier (AdmittanceID) is created. Since patients can have multiple admissions, a unique identifier is generated for each admission to the clinic.

Injuries are described in an abbreviated injuries table of common injuries that students will be familiar with, such as a laceration, broken bone, concussion, etc. Commonly used terms are placed in a char(40) field, rather than medical terminology, and an int is used as the primary key.

The medicines table is used to record medications that can be dispensed to patients. Each medicine is identified using a unique integer. The names of the medications are recorded in a char(25) field.

A patient admittance injury table is created to show the relationships between the patient, their admission and their injury. A composite primary key is used, comprised of the AdmittanceID and InjuryID. This intersection table is used to allow for the many-to-many relationship between admissions and injuries.

Medications given to each patient are recorded in the patient medicine table. It uses a composite primary key of the PatientID and MedicineID. Using the DateTime datatype of SQL Server 2000, the administration of a medication is recorded as AdminDate. For simplification purposes, all medications are treated as delivered in milliliter dosage increments. An effectiveness field is used to rate the patient's self report of their response to the medication, since nearly all of the medications are analgesics (pain relievers).

The symptoms table records the observed or self-reported complaints of the patient. Each common symptom is described in a char(40) field and identified by a unique int as SymptomID.

The patient symptom table is an intersection table reflecting the many-to-many relationship between the patients and symptoms. A composite primary key of PatientID and SymptomID is used.

Lastly, a billing table is used to record data on billing dates, and payment amounts and dates. A BillingID number is generated for each bill and is an int that serves as the primary key for this table.

MOAB MEDICAL CLINIC XML MODULE

When the students finish creating and populating the database, they are ready to start learning about XML and how to apply it to the MMC. The students will learn about the basic structure of XML, how to format XML on the web, the use of validation tools and how to bring the data into a database from an XML file. An example of part of the XML portion can be found in Appendix B.

We include links to several tutorials found in the XML Module [1]. Upon completion of these tutorials, the students will be given assignments designed to teach the different methods of accomplishing a given task, and then the student uses this new knowledge to complete a correlating assignment. The online tutorials come from several popular and well known computer companies, like Microsoft and Sun Microsystems. The tutorials coincide with the assignments, teaching the students how to 1) create the database; 2) create an XML document using SQL Server 2000; 3) attach an XSL-FO document; and 4) bring a XML document into an existing SQL Server database. There is also a section in the XML Module that references how to setup the Microsoft SQL Server to view XML in a query analyzer.

The first assignment involves creating the MMC database, as stated in the previous section. The rest of the assignments are interrelated and build upon each other. This is important because the XML module is designed to provide students with a solid XML experience.

The assignments were developed as part of the active learning method. The students will take what is learned in the tutorials and will then complete the assignment to reinforce learning and master the material. Because this module is being used in the advanced database course, the solutions are available upon request.

After completing the tutorials the students then must work on the assignments. Because the tutorials are all online, the students can reference these while completing the assignments, thus creating a reference base of future resources. As the students complete the tutorials about a specific technology, they should move to the correlated assignment before moving on to the next XML technology.

CONCLUSION

We have presented a comprehensive teaching case that demonstrates the integration of a XML module into an advanced database course. XML is becoming an increasingly important technology, yet few schools are teaching students how to retrieve, format and display XML data. Based on a sound relational database built in earlier assignments, students proceed through a number of tutorials that develop their familiarity with key components of XML technology. We believe that this method offers a couple of advantages. First, students are reminded of the importance of good relational design principles as they build the MMC database into SQL Server 2000. Second, students learn XML technology in the context of an advanced database class. Third, this instructional approach lays an effective foundation for later courses in web services development or web application programming, without losing the course's focus on advanced database management.

With literally hundreds of competing software packages, languages, operating systems, and networking technologies, we believe students need to integrate and apply new learning within the context of previous skills. XML then becomes a part of student's fundamental understanding of data retrieval and formatting, rather than just one more language or tool.

SUGGESTIONS FOR FUTURE RESEARCH

Future research in this area will involve assessing learning outcomes in core database technologies when adding XML to the course material, developing and evaluating parallel materials for a course based on the Oracle

DBMS and assessing the transferability of the domain data to other student populations, and evaluating the impact of XML on database security.

Figure 1: ER Diagram

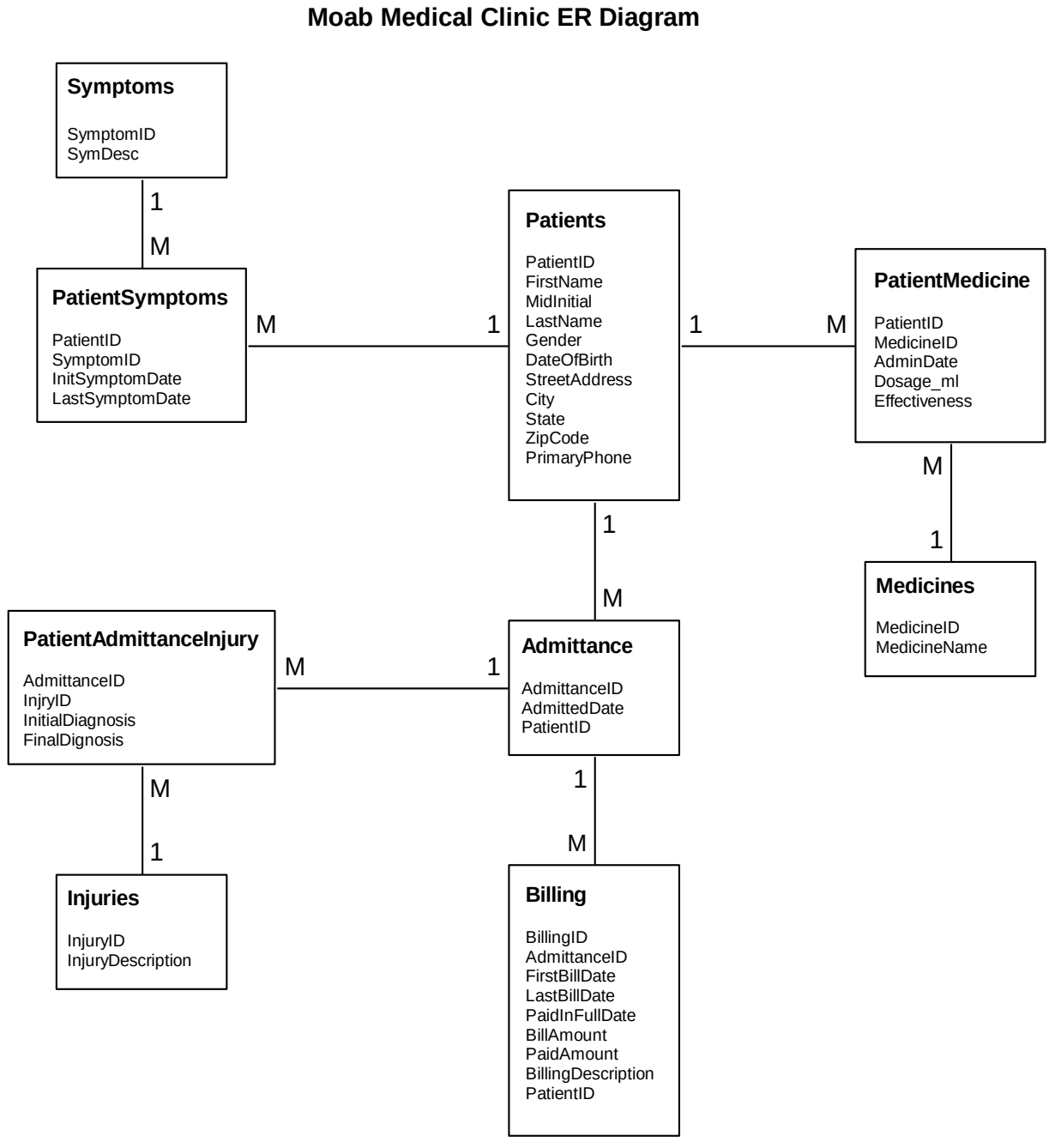
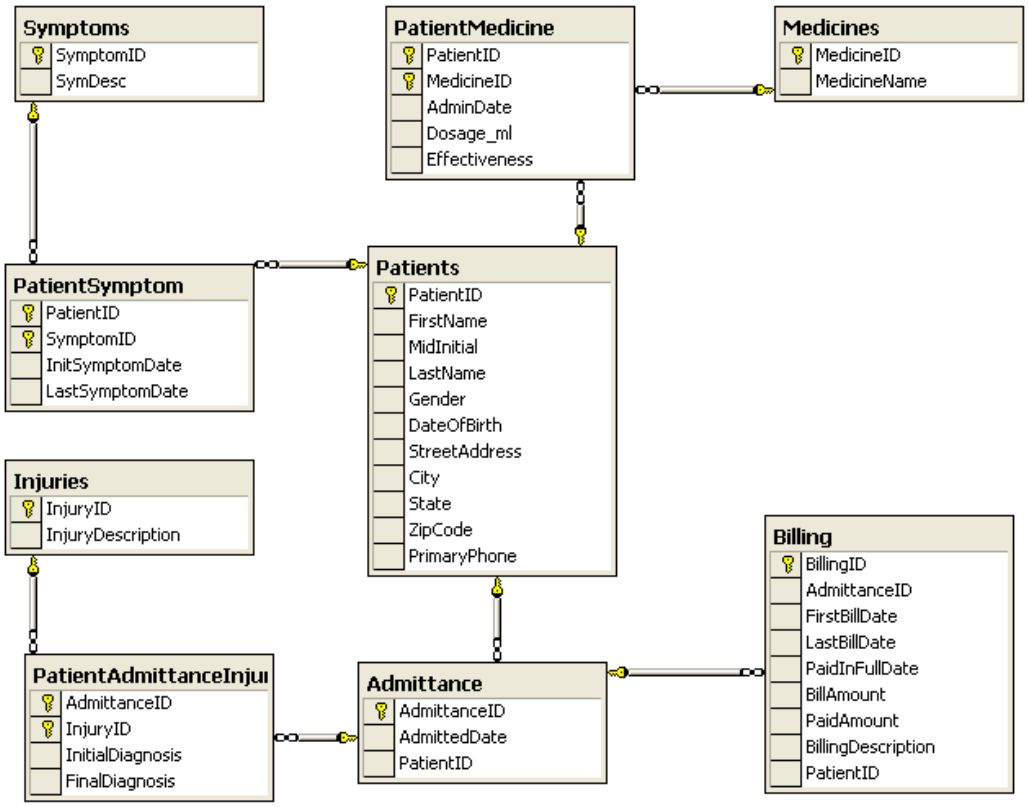


Figure 2: SQL Relationship Diagram



APPENDICES

Appendix A – Create the Moab Database Assignment

The Moab Medical Clinic (MMC) sees patients who suffer injuries while engaging in outdoor recreation. In this case, we are only concerned with a limited set of injuries – broken bones, concussions, lacerations, etc. Each injury is recorded separately, since a patient may have multiple injuries.

Patients present themselves and must provide biographical and contact information in order to be admitted to the clinic for treatment. Since it is possible to receive treatment on more than one occasion, each admission of a patient is recorded separately.

When a patient enters the clinic, the medical staff will record the patient's symptoms. MMC Staff will want to know how long these symptoms have been occurring, and the last time the symptom occurred.

The patient may receive medications to treat his or her symptoms. MMC staff will record the dosage in milliliters, and need to know when the medication is administered. A nurse or other MMC staff person will ask the patient to rate the effectiveness of the medication.

Create a database and name it MMC from within the Enterprise Manager. Step-by-Step instructions are available on the course website.

Determine the relationships between the following entities. Although other entities that could exist in this database, only focus on the entities described. One of the tradeoffs in database design is to capture all needed information, without recording information that can be retrieved from other sources or that is not necessary for the function of the database.

The entries below describe entity names and their attributes. Determine the appropriate datatypes and sizes for these attributes. Create each entity as its own table. Consider the relationships between these entities before you attempt to create your relationships. Which attributes can be used as foreign keys? Out of the candidate keys, select the primary key, which could be a composite key. Clearly record which attributes reference other attributes. Determine whether each attribute could be null.

Entities

Entity 1: Patients (5 tuples)
PatientID, FirstName, MidInitial, LastName, Gender, DateOfBirth, StreetAddress, City, State, ZipCode, PrimaryPhone

Entity 2: Admittance (8 tuples)
AdmitanceID, AdmittedDate, PatientID

Entity 3: Injuries (10 tuples)
InjuryID, InjuryDescription

Entity 4: Medicines (10 tuples)
MedicineID, MedicineName

Entity 5: PatientAdmittanceInjury (20 tuples)
AdmittanceID, InjuryID, InitialDiagnosis, FinalDiagnosis

Entity 6: PatientMedicine (20 tuples)
AdminDate, Dosage_ml, PatientID, MedicineID
Entity 7: Symptoms (8 tuples)
SymptomID, SymDesc

Entity 8: PatientSymptom (20 tuples)
PatientID, SymptomID, InitSymptomDate, LastSymptomDate

Entity 9: Billing (10 tuples)
BillingID, AdmittanceID, FirstBillDate, LastBillDate, PaidInFullDate, BillAmount, PaidAmount, BillingDescription, PatientID

Assignment Requirements:

1. Build the entities and relations in SQL Server 2000.
2. You must enforce the relationships.
3. Choose your data types consistently and appropriately.
4. Populate the relations as specified. Use reasonable data.
5. Draw an ER diagram that reflects the written case and the tables. You can use an ER tool, or a drawing tool.
6. Design and include 5 constraints.

Turn in the following hardcopies:

- A cover sheet with your identifying information
- Your ER diagram.
- Proof that you created the tables and proof of the data types.
- Proof that you created the relationships.
- Proof that you populated the tables appropriately.

Appendix B – Example of a portion of the XML Document Assignment

Part 1

The Moab Medical Clinic has decided to become completely electronic in the claims department. To do this they need to send the patient data to the insurance company in a form that any insurance company can read.

Create a query that pulls out specific information for a patient to send to an insurance agency for billing purposes. Needed information includes; Patient ID (SSN in real world), First Name, Last Name, Date of Birth, Gender, Bill Amount, Paid Amount, Billing Description, Final Diagnosis, Injury Description.

Part 2

The Moab Medical Clinic has been informed by several different insurance companies that they have stopped using the EDI standard and have moved to the XML Standard.

Using the same query from Part 1, change the output to XML, where the PatientID is an ID attribute and all the other fields are elements.

Appendix C – An example of the material covered in class "XML from MSSQL"

You are now the Insurance Company, and would like to bring in the XML document from MMC. Bring the data into the Insurance database, from XML.

For XML Raw

SQL Server has built in Transact-SQL statements to retrieve data out of an existing SQL Server database. There are three methods used; Raw, Auto and Explicit.
Raw is the easiest method to pull XML output. The result set is in attribute format and each record is started with the generic ROW tag.
Enter the following Select Statement into your Query Analyzer using the Moab table

```
SELECT BillingID, AdmittanceID
FROM Billing
FOR XML RAW
```

Run the query.

Notice the result set starts with the generic word <row>, and each entity is displayed as an attribute.

```
<row BillingID="1" AdmittanceID="1"/>
<row BillingID="2" AdmittanceID="4"/>
```

For XML Auto

Using the AUTO function allows for more control over the result set. To extract using the Auto function, simply add the statement 'For XML AUTO' after the last statement in your Query.
Let's try this. Enter the following Select statement into your Query Analyzer using the Moab Table.

```
SELECT BillingID, AdmittanceID
FROM Billing
FOR XML AUTO
```

Check the query by running it.

Notice the results are now in XML format and that the result set looks similar to the RAW method except instead of <row> it now uses the table name <billing>.

```
<Billing BillingID="1" AdmittanceID="1"/>
<Billing BillingID="2" AdmittanceID="4"/>
```

Note also, that the results are still in attribute centric mode, where all the columns are represented as attributes.

For Xml Auto, Elements

Using the Elements option, allows the result set to be viewed in element centric mode. Where all the entities are now elements.

Try this on the same statement as before, expect this time add a comma and the word ELEMENTS on the end of the XML statement.

```
SELECT BillingID, AdmittanceID
FROM Billing
FOR XML AUTO, ELEMENTS
```

Run the query and notice the differences.

```
<Billing>
<BillingID>1</BillingID>
<AdmittanceID>1</AdmittanceID>
</Billing>
<Billing>
<BillingID>2</BillingID>
<AdmittanceID>4</AdmittanceID>
</Billing>
```

Notice that all the entities are now in element centric mode, with <billing> as the parent element.

For XML Explicit

Using Explicit mode requires more complexity and syntax. Basically, each variable entity needs to explain to the SQL XML processor exactly what format to display it as. For example, if you wanted BillingID to actually be an attribute and AdmittanceID to be an element, you would simply list each as such, in the Select statement.
Let's give it a try. Use the same Select statement as before except this time add the following code to the end of each entity name.

```
SELECT      BillingID AS [Billing!1!BillingID],
            AdmittanceID AS [Billing!1!AdmittanceID!element]
            FROM Billing
            FOR XML EXPLICIT
```

If you were to run this query now, we would still be missing a very important part. At the beginning of the Select statement, we must add the statement '1 as Tag, NULL as Parent'.
Now the code should look like this.

```
SELECT      1 as Tag,
            NULL as Parent,
            BillingID AS [Billing!1!BillingID],
            AdmittanceID AS [Billing!1!AdmittanceID!element]
            FROM Billing
            FOR XML EXPLICIT
```

For the purposes of this assignment, just realize that in explicit mode this feature will eventually allow for multiple queries from multiply tables, distributed in a hierarchal structure in XML.
If we run this query, this should be the result set.

```
<Billing BillingID="1">
<AdmittanceID>1</AdmittanceID>
</Billing>
<Billing BillingID="2">
<AdmittanceID>4</AdmittanceID>
</Billing>
```

Notice that there is a mixture of attributes and elements, and that the root element is <Billing>.

REFERENCES

1. Marshall, B. 2004 XML Module. Available from http://olsen.usu.edu/Olsen/XMLModule/index.htm [accessed 1-05].
2. Wagner, P.J., Moore, T.K. 2003 Integrating XML into a Database Systems Course. Paper presented at 34th SIGCSE technical symposium on computer science education, Reno.
3. Sperberg-McQueen, C.M. & Bray, T. 1997. Extensible Markup Language. Available from http://www.cs.queensu.ca/achallc97/papers/p050.html [accessed 1-05].
4. Codd, E.F. 1970. A Relational Model of Data for Large Shared Data Banks. Comm. ACM 13:6.
5. Gorgone, J.T., Davis, G.B., Valacich, J.S., Topi, H., Fenistein, D.L. & Longnecker, H. E. 2002. IS 2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems. Available from http://www.acm.org/education/is2002.pdf [accessed 1-05].
6. Computing Curriculum 2001. Available from http://www.computer.org/education/cc2001/ [accessed 1-05].
7. SQL Server 2000 XML Overview. 2001. Available from http://www.microsoft.com/technet/prodtechnol/sql/2000/evaluate/xmlsql.mspx [accessed 1-05].
8. World Health Organization, International Classification of Diseases. Available from http://www.who.int/whosis/icd10/ [accessed 1-05].